Chapter 7 – Examining the Knowledgebase

Abstract This chapter will take a close look at the knowledgebase that is built up from the analysis process and how it maintains the important relations between multiple components in very complex systems. It starts with an explanation of the knowledgebase structure derived from the formal definition of system in chapter 2. A sample of knowledgebase tables and forms will be used to show how to capture the information about the system from analysis. We build a basic knowledgebase model using an existing database management tool to show how the indexing scheme works to maintain relations of components.

7.1 Why We Call It a Systems Knowledgebase

A database is an organized collection of data from which users make queries in order to get information[1]. The organization of the data is generally based on specific uses. For example, an accounting database contains a variety of revenues and costs[2] by time-date stamped transactions (debits and credits). Routinely the accounting system produces profit-and-loss statements, among other summaries, that tell the management how the organization is doing with respect to its operating and financial goals. This counts as information derived from data by data processing programs. The further advantage of a database management system is that one can formulate special or ad hoc queries that are not necessarily tied to the overall organization goals. For example, the Accounts Receivable department would like to know which of their customers are in the arrears in payments by more than three months (or any time range) in order to take appropriate actions to get the deadbeats to pay up. Marketing may pose an ad hoc question about customers that had purchased the most of a product, or how much of a product had been sold in the last year. A database management system (DBMS) allows such questions to be formulated and processed.

The example of the use of a DBMS to hold accounting data shows how the accumulation of data can be *a priori* organized to produce information. But should we call the data knowledge?

By itself, the data cannot be considered knowledge. It is the organization of the 'schema' of the database that makes it useful in the generation of information. The structure supports the efficient use of the data in a rule-based way that serves decision making. The Greek-derived word, *episteme*, conveys this concept. According to Dictionary.com the definition of episteme is: "a system of understanding or a body of ideas which give shape to the knowledge of that time".

[1] For background see the Wikipedia article: https://en.wikipedia.org/wiki/Database. Accessed 12/6/2017.

[2] In the form of things like Accounts Receivable and Accounts Payable, among others.

The key concept here is the 'giving shape' to the knowledge[3]. Episteme, in our thinking, refers to the structure (or the 'system') of the body knowledge and not just the 'facts' or data. That structure has to capture the actual, functional relations between the facts and data. So in the somewhat limited world of accounting, the database management system and the schema of the data as the 'books' of the organization make the contents knowledge of the current state of the financial situation.

This will be the argument for systems knowledge. We organize system facts into a schema for system structure. In fact that organization is based on the set of equations in Chapter 3 that collectively define a system and its components, especially in relation to its environment.

7.1.1 The Ontological Status of Knowledge

In Chapter 2 we made a bold claim, that something we called "Knowledge" was a first-class ontological substance. We equated knowledge with the structure/organization of material forms. And we argued for how knowledge could change as a result of the flow of information. To recap[4], when a system receives a message, the encoding of message states is accomplished in the modulation of energy levels per unit time and messages are conveyed serially (i.e. as a time series) through channels or broadcast media. The encoding is done by the sender. Upon receipt those energies may be amplified by virtue of the receiving system being *a priori* prepared to interpret the meaning[5] of the message; that is, all of the machinery for reacting to the causal influence of the message is already in the receiver[6]. The amplified energy flows contribute to additional work on the receiving system's structure, it is altered in such a way that future messages of the same kind do not result in the same amount of work; the receiver becomes more dissipative as a result of the changes in structure resulting from the work done on it. That change constitutes a change in knowledge, or the preparedness of the receiver to deal with future energy flows.

Thus, knowledge, like information, is not separate from matter and energy. Rather, it is encoded into the systemic structures of matter that allow energies to flow from high potentials to low potentials with a minimum amount of disruption to the structure of the system. Knowledge,

[3] The addition of "of that time" seems redundant in that all we have at any point in history is the knowledge of that time.

[4] For an extended explanation see Mobus & Kalton (2015), section 7.2.1.9 Codes, pages 274-275. Chapter 7 is devoted to explaining Information and Knowledge and their relations.

[5] As explained in Mobus & Kalton (2015), chapter 7, meaning is based on how a receiver interprets the information in the message.

[6] Senders and receivers co-evolve over time owing to the history of messages sent and received. As such the receiver is a priori prepared for the kinds of messages (i.e. the average symbols in a message) that the sender might send. When the sender sends a message that has a low probability, this triggers a reaction (work) in the receiver so as to modify its own structure (Mobus & Kalton, 2015, Chapter 7).

as structure, makes a system more resilient, stable, and sustainable over time and fluctuations in the ambient conditions.

Informational messages report changes that have occurred in the receiver's environment and results in the receiver altering its own structure such that future messages reporting the exact same changes is less informational. In other words, the first time a specific message is received reporting (via its encoding) a change, the degree of that change corresponds to the amount of information in the message. If the message is repeated (after the receiver has, in fact, changed its own structure accordingly) then no information is conveyed and no additional changes in structure are done in the receiver.

The claim, thus, is that the actual structure of any system at a given point in time constitutes its knowledge of what to expect from its environment. This is why both information and knowledge are characterized mathematically in terms of probabilities of states of systems (either the sender or the receiver). A receiver expects that communications from its environment (linked through message interfaces across its boundary) will confirm its own encodings of likely states of the entities from which it receives messages, as in "I'm still in state X". That kind of message has a high likelihood of being received by virtue of the fact that it has been the most frequent message previously received[7]. Receipt, in the next time frame, of "I'm now in state Y," had a lower expectancy and so conveys more information. The receiver has to alter its own state to reflect this situation, that is, to increase its expectancy that future messages will more likely be about state Y than state X, hence we say that the receiver has learned something about the sender, namely its new state.

In a dynamic world, where systems are continually changing their states and sending messages to other systems, the flow of information and the generation of knowledge is forever operating. Indeed, since systems of sufficient complexity can adapt and evolve (particularly the latter) we assert that knowledge and information are that which grows in amount as the Universe evolves. That conjecture begs for a proof.

7.1.2 Knowledge Accumulation

Systems that have mechanisms for combating the effects of entropic decay in their structures (i.e., are autopoietic) are able to encode and maintain knowledge as it is gained from the work generated by the receipt of information. They actively work to not forget what they have learned. The human brain is exemplary in the animal domain. Our memories encode not only the current states of the world, but our capacity for episodic memories constitute a record of the time series of events that led from prior states to the current state. We remember the stories of how things

[7] More explanation for the process of representing expectations and changing them as a result of informational messages is given in Mobus & Kalton (2015), chapter 7, section 7.3.3, pages 289-296.

got to be the way they are. We remember a history even if imperfectly[8]. Human long-term memory results from the way in which neurons maintain synaptic strengths along with circuit designs that periodically refresh the stimulating inputs that promote excitability in specific pathways (speculated to be one of the functions of dream sleep) of memory traces that have proven useful in mental activities (i.e., have resulted in rewarding feedback)[9].

Computing systems employ a hierarchy of memory devices that range from highly volatile memory cells (CPU registers and main memory) to recordings of state information in magnetic domains on a quasi-permanent substrate (hard disks and tapes). While the computer is powered, the memory system does considerable work in maintaining the states of memory cells. When the power is turned off, these dissipate and the overall state configuration is lost. The genius of modern computer design, however, is that the state of the volatile components can be preserved in the magnetic substrates between power states so that the long-term memory preserves the last state of the working memories. Even these preserved recordings are subject to entropic decay effects, but over extremely long time scales relative to human interests. And, with the right sort of maintenance mechanisms in place (periodic refreshing), as with autopoietic mechanisms in living systems like the brain, knowledge encoded can be preserved indefinitely, at least in principle.

The pace of scientific discovery is higher today than at any time in history. The accumulation of knowledge is staggering. But accumulated knowledge is only useful knowledge if it can be readily retrieved (see below). The rate of accumulation, especially with the traditional means of storage raises severe issues.

In 1956 Kenneth Boulding recognized the need to establish a more holistic systemic way to manage knowledge, then obtained by the sciences using reductionist methodologies, for the most part, and maintained externally from human memory, i.e. the library. Speaking to the architecture of knowledge storage and retrieval mechanics, he observed:

> … the problem of the adequate descriptions of complex structures is still far from solved. The theory of indexing and cataloging, for instance, is only in its infancy. Librarians are fairly good at cataloguing books, chemists have begun to catalogue structural formulae, and anthropologists have begun to catalogue culture trails. The

[8] It must be noted that while it is true our memory systems allow us to capture and maintain a trace of history as we perceived it happening, those same systems are notoriously faulty with respect to fidelity of details, even, often, actual sequences of events. For example, we now understand that eyewitnesses to crimes all too often make faulty reports about perpetrators (e.g. identifications in police line-ups) or events (as in sworn testimony in court). The source of these faults can best be understood when we realize that our memories are not recordings in the same way that, for example, a digital recording of a movie is done. For a very excellent overview of the nature of memory in the human brain by two of the premier investigators of the phenomena see, Squire & Kandel (2009).

[9] A simple example is the procedural memory capacity for riding a bicycle. Though no one knows yet quite how this is accomplished the human brain retains the motor memory for doing so even when long gaps in riding occur. Contrast this with memory aspects in much more primitive brains where conditioned responses decay over time (see: Alkon, 1987 for seminal work on nudibranch memory).

> cataloguing of events, ideas, theories, statistics, and empirical data has hardly begun. The very multiplication of records however as time goes on will force us into much more adequate cataloguing and reference systems than we now have. This is perhaps the major unsolved theoretical problem at the level of the static structure. (Boulding, 1956, 296-297)

Boulding (ibid) noted further that progress toward a solution was being made and also noted that this applied to the kinds of instruments scientists were employing to obtain knowledge as well.

In 1956 our methods for cataloguing and indexing knowledge were indeed clumsy, as toddlers are clumsy when they take their first steps. The modern era of computer-based systems and massive capabilities as offered by Google and other masters of storage and retrieval offers the opportunity to establish a much more systematic approach to knowledge management. Several factors need to be addressed.

7.1.2.1 Knowledge Encodings

Knowledge encoding refers to the mechanics of changing system structures to reflect the prior receipt of information. There is a significant difference between the way the brain and a computer encode knowledge regardless of the storage mechanisms. Even so, early research on the differences in the methods points to promising ways to translate knowledge between the two. At this writing there are several efforts to emulate human knowledge encoding (as in neural circuits) in digital forms. In principle this is not unexpected. But it is not a trivial problem either. Meanwhile several research groups are tackling the problem of how to do in computers what brains do in real life (Hawkins, 2004)[10].

The basic problems involve representation of knowledge and its storage/retrieval. In computer science the latter aspects have been dominated by algorithmic approaches to efficient methods for doing these tasks. In the history of computer science, the methods have been motivated by a straightforward notion of representation. Namely, a data item is simply encoded in bits stored in memory at any level in the persistence hierarchy. For example, an employee's number, name, pay rate, and other relevant data are encoded using binary representation codes like ASCII[11] to directly represent the data. The data are then organized into a relation called a record that can be efficiently written to an external medium (hard disk file). Retrieval involves finding the data given a key, like the employee number, which might involve a search through a long list or, more sophisticatedly, using something called a hash code for the employee number to go directly to that record in the file.

[10] See, for example, Wikipedia, The Blue Brain Project: https://en.wikipedia.org/wiki/Blue_Brain_Project. Accessed 10/17/2017.

[11] American Standard Code for Information Interchange. See the Wikipedia article: https://en.wikipedia.org/wiki/ASCII for more information.

Even though the brain doesn't operate in this manner, the speed of computation has reached a point where the search methods of the algorithms competes favorably with the capabilities of the brain (which performs a massive parallel search, but at the slow rate of neural signaling!)[12] There is, today, a good reason to believe that even though the computer operates on different principles with respect to storage and retrieval of data, that it has the capacity to emulate the brain such that it becomes a powerful mechanism for capturing, storing, and retrieving information beyond the capacity of any one human brain. This is the basis for claiming that a computer-based knowledgebase can be an important tool in systems understanding.

7.1.2.2 Knowledge Representation

A fundamental problem in using computers to represent system knowledge in a way that allows ready use of that knowledge (i.e. recall and processing) has been the methods of representation, or how knowledge is organized in a memory structures in a digital system.

In Chapter 3 we presented a formal definition of system. We also claimed that this definition is based on an ontological primacy of the concept of systemness that, through the evolution of intelligence and consciousness has resulted in our (humans') capacity to think in terms of systemness. The language of systems was claimed to be based on this fundamental idea.

What we assert here is that the representation of knowledge, which is at base about the systemness qualities of objects in the real world, must be based on those qualities. In other words, knowledge of anything should be based on the definition given in Equation 3.1. In this chapter, we derive a knowledgebase structure, based on that definition, to be captured in a computational structure, emulating system knowledge. We present, here, a computational method for encoding and representing system knowledge as it is derived from the analysis procedures presented in the prior two chapters. The importance of this representation will become much clearer when we tackle the problem of generating models (Chapter 9) and how those models impact our grasp of policies (Chapter 12) and system designs (Chapter 13). In this chapter we will use Equation 3.1 (along with its derivatives in Chapter 3) to create a database schema that will be a usable representation of system knowledge about any kind of system. The result of this representation will be ready access to system knowledge and especially, the generation of system models for understanding, design, and policy generation.

7.1.2.3 Knowledge Retrieval

The structure of how knowledge is stored in a persistent medium plays an important role in how accessible it is for retrieval and use. Think of the library.

There are many forms of stored knowledge, different formats and media (e.g. books, journals, microfiche, etc.). All of these are referenced through an index system (as noted above)

---

[12] Additionally, the speed of modern computer hardware is so great that it would seem to more than compensate for the slowness of biological neurons. The jury is still out on that issue, however.

that records key aspects of the knowledge (e.g. titles, authors, dates, etc.) and a location code so that one can go to the location in the library where the material is stored. Assuming that it hasn't been checked out, the researcher can then physically access the knowledge.

The library stores, in an organized way, what we should call potential knowledge. By our definition of knowledge in Chapter 2, knowledge is a structure but it is also relevant to an effective process (causal). The knowledge structures in a library are non-effective until they are transmitted into the minds of readers and cause change in those minds. The researcher has to retrieve and read the resource in order for it to produce knowledge in their minds as actors and agents. The retrieval process is actually informational with respect to the researcher. The knowledge potential stored in the resources and organized in the library does not actively suggest itself to the researcher.

The methods of indexing the resources was primitive. Once, not long ago, the constrained information about the resource, title, etc., which could be called 'meta-knowledge' was available on a cardboard index card in a file system. An indexing scheme, like the Dewey Decimal system was used to aggregate collections according to subjects (then sorted in authors' last name order). One needed to first understand how the index structure was segregated by subjects and which leading index codes related to which subjects[13]. Then one had to guess as to what the contents of a resource might be, unless one were pointed to the resource by name from another resource (the bibliographies and references, for example, in journal articles or books). In the former case one had to be willing to pay the cost of effort in retrieving the resource, reading it, and then determining whether or not it was relevant (produced knowledge in one's brain). In the latter case one could retrieve the work but probably would be using it more to confirm already gotten knowledge, i.e. it would not be informational.

Today we still basically use the same system but now our index is managed in a computerized system. This newer system permits a bit more information to be supplied within the index itself, such as an abstract of the work. An even better method for finding relevant work is supplied by on-line journal access. One still needs to know which journals (by name) contain articles most likely relevant to their interests. But the search for, and preliminary information about the work has been streamlined, as well as has the retrieval process itself. Today a disciplinary researcher, in many fields, need not actually visit a library except to pick up a book they ordered from the on-line access platform.

With the advent of the technology of search engines the retrieval of resources has become more automated. The researcher can now aggregate a set of what are key words to them but

[13] As much as the division of knowledge by academic departments and their rules for junior professors to gain tenure by narrowing their research to very specific topics (and the complicity of journal publishing) the Dewey indexing scheme is a culprit in creating knowledge silos. No one is to blame for any of this, of course. It was a natural consequence of trying to come up with a means to manage knowledge storage and retrieval in a 'systemic' way.

might not necessarily have been identified as such in the source document. Rather it is the words themselves from the body of the text that are searched and indexed in a database of resource addresses. Using a web browser, the researcher can google[14] their words of interest and get back a list of resources that contain those words (in logical relations such as AND or specific order).

Even so, with all the technological improvements in search, what have we actually gained? The fundamental problem of canalization, keeping knowledge segregated in disciplinary silos, still persists. Knowledge has become compartmentalized in a way that makes interdisciplinary inquiry increasingly problematic. While a search engine might bring us documents from a wide variety of subjects based on the key words we put together, they can also bring a flood of documents that do not necessarily report on transdisciplinary work[15]. On the other hand, the rise of interdisciplinary research, driven by the need to understand systems more broadly, has begun to generate resources that address this need. There are now journals, for example, that cut across disciplinary topics to reflect the real needs for understanding from a more generalized perspective. This is a positive trend, so long as the quality assurance mechanisms, like peer review, are upheld strenuously. But it does not address the larger problem of how to organize knowledge more globally so that it is readily available from any number of perspectives. The usefulness of knowledge depends, ultimately, on how it can be retrieved not just from one perspective, no matter how interdisciplinary it might be, but from all perspectives.

The knowledgebase aggregates knowledge, structured by the definition of system, such that it is available from any desired perspective. As an example, consider the new field of bioinformatics[16]. As the name implies it is an attempt to organize biological data such that it can be retrieved and used from multiple perspectives (e.g. biological, engineering, statistical, and others). This is a step in the right direction. Data has to be organized by a semantics that is general in order to be accessible thus. On the other hand, if it becomes too general, that is simply accumulated such that anyone can access it from anywhere the burden is on the intelligence of software to make it accessible and impose some particular perspectives semantics on it[17]. This is also the problem with the World Wide Web and why search engines are only a partial aid in making sense of the distributed data. Bioinformatics starts with a more structured database schema based on general biological understanding. For example, genetic sequence data can be

---

[14] It is interesting how words can morph in the language. What was the name of a search engine company, Google, has morphed into a verb describing the action of making a search.

[15] It is incumbent on the researcher to be very thoughtful in choosing key words in order to minimize the flood. Fortunately, most search engines use some form of AI content analysis to rank returns according to what the AI "perceives" as the user's interest.

[16] See the Wikipedia article: https://en.wikipedia.org/wiki/Bioinformatics. Accessed 11/3/2017.

[17] Data mining using statistical methods to discover patterns in large data sets is an example. The data schema for these sets are based on particular uses such as a customer database for keeping track of, say, accounts receivable. Marketing can use mining on historical data to look for trends *a posteriori*. See the Wikipedia article: https://en.wikipedia.org/wiki/Data_mining.

readily linked with cellular and tissue development, and that with biological functions such as protein activity, etc. Even though there are specific uses (research questions), say for example, searching for specific gene sequences in a species' genome. The use can come from a variety of perspectives that have in common the concept of genetic inheritance, e.g. proteomics (proteins derived from genes), genealogy (tracing heritage), or evolutionary biology.

7.1.3 Knowledge Use

Knowledge is not knowledge unless it is useful! And, ultimately, that use comes from having a perspective and asking a question. For example, in the case of bioinformatics just described above, an evolutionary biologist may be able to discern the difference between convergent evolution from a case of dispersed species (Losos, 2017). Significant new insights into the dynamics of evolution that indicate it is not strictly a historical happenstance have been obtained through examining the cladistics of various species that resemble one another but turn out to be derived from very different last common ancestors (convergent evolution).

Data becomes usable when it is transformed into knowledge by: 1) being stored in structures that are amenable to multiple perspectives; 2) having mechanisms that are able to generate constructive views based on those perspectives; and 3) providing answers to questions posed by the multiplicity of perspectives interested in what it all means. In other words, a knowledgebase is only called that if the underlying data are readily transformable, first into informational messages, and then into mental representations that affect agent actions. They support decisions among researchers about what to explore next.

The knowledgebase that we accumulate represents our best understanding of any system we analyze. This can be tested. From our knowledgebase we can generate models of the system at any desired level of abstraction, including models of subsystems (at lesser levels of abstraction and more detail – see chapters 12 & 13).

It is important to understand that this is what we are already doing in the disciplinary sciences. What we are not doing is generating these models from a systems-based knowledgebase. Rather, we generate models of systems from our ad hoc and fragmented knowledge of systems of interest based on disciplinary details. This process has served us well enough in the disciplines, when what we were interested in was how specific systems worked. But as we are increasingly concerned with larger and more complex systems of interest, say how human social systems interact with the environment, the traditional approach breaks down.

What we seek now is to grasp large-scale, complex adaptive and evolvable systems like the whole human social system in the supra-system Ecos. And the normal science process is not geared to tackle that kind of problem.

Alternatively, if we start to collect and organize knowledge of the world in a true (i.e., systemic) knowledgebase, then that knowledge will become more useful for our purposes. Our claim is that by recognizing the universal patterns of systemness across disciplinary knowledge domains, we can more effectively formulate questions relevant to any discipline but more importantly any transdisciplinary inquiry.

7.1.4 The Knowledgebase Structure Fulfills These Needs

The argument advanced here is that a database schema based on the structure represented in Equation 3.1 (and the subsequent equations in Chapter 3) provides a way to meet all of these requirements. It provides a mechanism for converting information gained from systems analysis into a knowledge structure (accumulation). It provides the template for how knowledge is to be encoded in a digital medium. It shows how to represent knowledge in a consistent, systemic manner. It provides a ready mechanism for retrieval of system knowledge in the form of maps, trees, and other mathematical representations that lend themselves to building models and generating designs. It is, in other words, imminently useful. Recognizing how the phenomena (things doing stuff) we examine are systems and capturing their parts and relations based on our definition of system provides us with a universal basis for understanding.

As importantly, the technology of structured knowledge storage and efficient retrieval is at hand. Consider two examples of massive, on-line methods in popular use today.

7.1.4.1 Google™ Products, Especially Efficient Search on a Dynamic Graph

The enterprise that has caused us to make a verb out of a noun that didn't even make sense when its first product was introduced has a whole constellation of products based on the Internet, client-server platform now called the "cloud[18]". Its original product was an on-line search engine that canvased the World Wide Web (WWW) for linkable files, indexing every meaningful word and phrase[19], and then providing extremely clever algorithms that let users formulate searches for documents based on those key words and their combinations. Over the years, with massive amounts of research into tools, such as artificial intelligence, that product has achieved a tremendous capability for finding many different kinds of resources in what for most seems like the blink of an eye.

Today the Google search algorithms work hard to report back the most relevant resources to a user. Not that long ago, a user might have to scroll through hundreds, if not thousands, of links trying to establish the relevance to themselves just from the titles. But by maintaining an internal knowledgebase of relations between resources, using history data on past retrievals that users actually retrieved, and other hidden but systemic mechanisms (including extensive data on the

[18] Cloud computing involves constructing massive server farms (banks and banks of powerful server-grade computers) with high bandwidth communications to the Internet. Not only is data stored but various applications such as word processing are also available making it unnecessary for individuals, or even companies, to own their own copies of software. See the Wikipedia article: https://en.wikipedia.org/wiki/Cloud_computing for more details.

[19] Meaningful in the sense that they are not common or helper words, like 'the' or 'and'. Meaningful words are cues to the meaning of the content of a document. They are often singled out as 'key' words, for example in scholarly texts.

users themselves) the likelihood that the top five links reported in a search are going to be exactly what a user was looking for is quite high.

One major problem with resources stored in the WWW, as well as social media platforms, is lack of verification or critical review of the contents. Librarians spent no small amount of their time filtering resources and verifying legitimate content for non-fiction work. Google has tried to mitigate this problem in the electronic medium to some degree by classifying some content by categories that have various levels of reliance. For example, Google Scholar™ handles scholarly articles linked to reputable journals and researchers. They have imposed more useful organization on the mass of knowledge that helps to make it truly knowledge.

Another very popular product for Google is Google Maps™. The company has linked cartography, GPS, and a massive database of locations such as restaurants and gas stations so that users of smart phones, for example, can retrieve information about where they are, where they are wanting to go, and what they will find when they get there. The product is truly a wonder of computing technology usage. At the time of this writing, according to the Web site, Mashable, "Combining satellite, aerial and street level imagery, Google Maps has over 20 petabytes of data, which is equal to approximately 21 million gigabytes, or around 20,500 terabytes[20]". That is some serious data storage! Recall from the last chapter that we introduced the idea of applying systems analysis to the human social system and started by thinking about how it is a subsystem of the whole Earth. Google Maps seems to have started laying the ground work for that project.

7.1.4.2 Wikipedia: Hyperlinking and Collaborative Documentation

As the reader will have certainly noted by now, we frequently refer to Wikipedia articles for background information. This is because the largest, most comprehensive, though not always reliable, encyclopedia the world has ever known is on-line and free (with a caveat emptor proviso)[21]. Our use of this medium is predicated on recognizing that topics that are technical in nature, i.e. science and engineering, have proven to be quite reliable over years of usage. And the articles tend to improve over time.

A wiki is a special kind of WWW application that allows multiple users to edit pages in order to support a community of practice (and knowledge) to capture their expertise for access by all. Wikis are in widespread use for collaborative projects, but the Wikipedia, and its sister

---

[20] See: http://mashable.com/2012/08/22/google-maps-facts/#HLcHq8vZ9Zqw

[21] Wikipedia has an article about itself: https://en.wikipedia.org/wiki/Wikipedia:About. The service is free to users and ad free as well. For this to continue the Wikimedia Foundation (https://en.wikipedia.org/wiki/Wikipedia:Wikimedia_Foundation), which administrates the encyclopedia and is a non-profit organization, does request donations from users who have found it a valuable resource (the Public Radio model of funding). This author does make an annual contribution to support the continued access to information from books like this.

projects, such as WikiBooks and WikiQuotes, use the same wiki platform to accumulate and organize the contents of a wide variety of knowledge domains.

One of the most powerful aspects in wiki technology is the use of hyperlinks embedded in the pages, which enable researchers to rapidly look at related information. A wiki page can be edited by a collaborative group. The Wikimedia platform has developed an extremely sophisticated set of tools for making this efficient for distributed contributions. The structure of a Wiki page ensures that the 'right' information is easily found on the pages.

As with Google, Wikipedia depends on the server farm technology in order to serve a world-wide audience[22] of knowledge seekers on a 24/7/365¼ availability schedule. Thus far, the Wikimedia Foundation has done an exemplary job of keeping this massive resource available.

What Wikipedia represents is a mechanism for organizing knowledge in a way that makes it efficient to retrieve and useful to knowledge seekers. And it does this on a world-wide scale.

7.1.4.3 A Google-Wikipedia Mashup

What Google products and Wikipedia demonstrate is the feasibility of capturing and organizing massive amounts of systems knowledge. What we are suggesting here is that using the definition of system from Chapter 3, designing a knowledgebase schema, and applying technologies similar to these two approaches provides a technical solution to how to capture, store, and retrieve the kind of knowledge needed to understand complex systems, specifically CAESs. Both organizations have aggregated experience in this problem space which could be applied to solving the problem of developing a system knowledgebase technology. They have proven that the volume of data is not really the problem. It is the organization of that data that is the key to retrieval and usefulness. In what follows, we present a preliminary vision of how system knowledge could be organized for the purposes of better understanding the systems of interest. That organization, along with the mechanisms for retrieval, provide us with the power to generate, for example, system simulation models and, where appropriate, system policy recommendations. Not only will we use aspects of distributed storage ala a Google-Wikipedia mashup, we will structure that storage around a core technology, the relational database model.

7.1.4.4 A Database Backbone

Google's approach to knowledge storage is to provide a massive index mechanism while leaving files and documents distributed. The company does provide cloud-based services for storage for specialized data. But its main talent is in retrieving files stored in Web servers, creating content-based index keys, and then retaining the URL in its own databases so that users distributed anywhere in the world can retrieve the files of interest based on these linkages. Because web pages and other linkable files are constantly being created, modified, and destroyed (or lost) the indexes have to be continually updated. Google has developed extremely efficient

[22] Wikipedia is also published in a large number of non-English languages.

algorithms for conducting regular reviews of previously indexed documents as well as discovering new documents within days (sometimes hours). Thus the searches for highly desired kinds of documents are usually up-to-date within a reasonable time frame.

The core of all of its technologies is sophisticated database storage. We will suggest a global knowledgebase system using a database to do something very similar, store content-based index information while allowing the relevant data to be stored in any appropriate form, such as active web pages. We will also use a wiki architecture to provide a rich hyperlink structure that can be edited by knowledge analysists as they capture system knowledge.

The major aspect of our database is a schema based on the definition of system from Chapter 3. We will organize the data in terms of the various characteristics of systemness. Recall from section KNOWLEDGE_USE above that what makes knowledge just that is its organization and retrieval. By organizing data about any system in the system schema, we make it useful and retrievable from multiple perspectives. Thus, we ensure that what we might have formerly called mere data, is, in fact, knowledge that we can use.

7.1.4.5 And a Wild Speculation

Is it feasible to design a global system knowledgebase?

Recall in the last chapter that we sought to establish the context of considering the human social system, the HSS, as a subsystem of the supra-system Earth. Consider a possibility. If we treat the Earth as the “master” SOI (with the sun, planets, stars, and other space stuff as the environment) then the master index of that system would be 0. Each of the major subsystems, the lithosphere, atmosphere, hydrosphere, etc., would be 0.1, 0.2, etc. The HSS, as unique animal-based subsystem, might be considered equivalent to any of these and be given the index, for example, 0.9. The economy, as a subsystem of the HSS would be indexed as 0.9.5, say. We will actually consider this in greater detail in the next chapter.

There is reason to believe that all of system knowledge, which means human knowledge of the world, could be relegated to a truly global knowledgebase. The Internet provides a basic communication fabric, the WWW a basic communication protocol, Wikipedia a basic architecture, and a Google-like database index to tie it all together and provide the core structure based on systemness.

Imagine wondering what the term for the power plant in cell metabolism is (mitochondrion) and doing a search for a standard system concept – power capture and conversion, a concept applicable to all systems – along with the words ‘living cell’. A Google search of “power capture and conversion” + “living cell” produced, among several, the web site: Molecular Biology of the Cell. 4th edition: https://www.ncbi.nlm.nih.gov/books/NBK26882/ , accessed 11/3/2017. That site provides very good information regarding power capture and conversion for animal cells (metabolism). Another site: https://www.scientificamerican.com/article/plants-versus-photovoltaics-at-capturing-sunlight/ , looks at a comparison of photosynthesis vs. photovoltaics, an interesting topic perhaps, but not cogent to the subject of how living cells capture and convert energy from sunlight per se.

Imagine further that you are curious about all the different kinds of such devices found in living organisms.

The designers of the Internet sought an architecture that was extensible but structured so that a message sent from one computer could arrive at a destination computer anywhere in geographical space by virtue of a routing system and an address system that gave a unique identity to both devices and a way to get the messages from one to the other efficiently (as well as robustly able to follow alternate pathways when needed).

7.2 Capture and Storage of the Relevant Aspects of a System

Recall the forms we filled out in Chapter 5 as we went through the top-down decomposition of the model system? This knowledgebase is where that information got stored. In Chapter 5 we used the definition of a system (from Chapter 3) along with the language of system (also from Chapter 3) to guide the procedure of investigation of the system's internals, turning the opaque box into a transparent box all the way down to the nitty-gritty details of the lowest levels of components – the atomic[23] components. As we did so, we captured the various kinds of system elements at each level of organization, assigned them codes and names that would give them unique identities and locations in time and space relative to all other elements. As we entered the data into the forms it was cross checked for duplication, syntax, lexical correctness, and completeness. And when issued an all-clear that data was entered into the knowledgebase.

In this chapter we will expose the internals of the knowledgebase in terms of how it is implemented in a relational database structure, provide some insights into the ways in which the entry process leads to the cross-checking, and provide some additional insights into how the knowledge, once captured thus, can be accessed to answer queries relevant to the particular system having been analyzed. This includes using the knowledgebase to generate model systems, design specifications or policy recommendations where appropriate. We will only be providing a peek at how these are accomplished from the inside of the knowledgebase structure in this chapter. In Chapters 9, 12, and 13 we will use the mechanics of generation to extract models, specifications, and policies to show how the whole system works.

Figure 7.1, below, shows the basic architecture of a knowledgebase (KB) system. The RDBMS contains the basic data derived from systems analysis according to Equation 3.1 (and subsequent equations in Chapter 3. Wiki pages that expand, augment, or otherwise enhance the data stored in the database are created by users during analysis (and other operations). Links (index links in the figure) to relevant pages are stored in the database for rapid retrieval. For

[23] Remember, the term "atomic" in this context does not mean element atoms. It means the smallest component that does not require further decomposition.

example, when an interface protocol is specified (a protocol wiki page) it is put in the wiki structure and a direct link to it is stored with the protocol slot in the RDBMS (see Figure 7.1). A core engine interfaces with the users (analysts and modelers) as well as the search engine and the wiki pages.

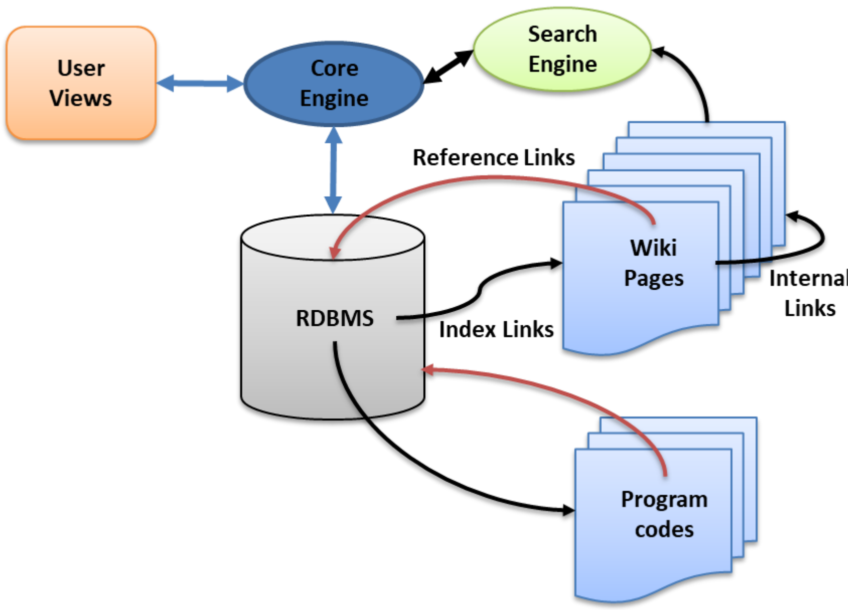

Fig. 7.1. The knowledgebase architecture includes an RDBMS with a schema based on Equation 3.1, multiple sets of wiki pages containing descriptive and relational texts (and graphics, etc.), and a search engine supplementing the wiki pages. The 'core engine' ties everything together and produces the user views (DBMS tables, wiki pages) that guide the analysis and control the retrieval of system knowledge. The KB core engine connects with the modeling core engine – see Chapter 9.

The wiki pages contain internal links that allow rapid traversal of relations in the page views. The search engine augments these links by creating a second form of indexing of other significant words in the wiki pages, such as domain-specific names of entities and processes. The core engine can also retrieve and format views such as maps and trees.

7.2.1 A Relational Database Schema for the Knowledgebase

Given the system definition in Equation 3.1, we will develop a set of relations (tables) in an RDBMS (relational database management system) that are based on the set of equations from Chapter 3[24]. The RDBMS schema captures the formal definition that gives structure to the

[24] As often happens in this fast-paced world of technology, shortly after this chapter was written the author was introduced to a new(ish) kind of DBMS called a "network database system" which represents data models as networks of nodes and relations. This, of course, would be ideal for representing the knowledgebase of a system since it is exactly a network of components (and a subnetwork of sub-components, etc.). However, after a survey of network database systems and their capacity to represent the hierarchical nature of networks we decided to stick

system knowledge, which, in turn, organizes the disciplinary knowledge for a particular system. In Chapter 6 we saw how the use of the language of system guides the collection of data for particular systems. In Chapter 8 we will demonstrate in even greater detail how that collection results in a knowledgebase of said particular system (the economic system). Now we will demonstrate how the knowledgebase architecture is used to accomplish this, from the insides.

This chapter, in essence, may be viewed as a kind of specification for the architecture of a systems knowledgebase.

You have already seen the forms that are used during analysis so the table structures, called schema, presented here should not be too surprising. In a sense, the forms simply reflect the schemas. What this chapter will do beyond repeating the data capture process of Chapter 5 is show some of the internal operations used in an RDBMS to store, retrieve, and process the data captured as part of verifying the different quality measures we introduced in Chapter 5. Chief among those was data consistency. We used that property to ensure that we did not leave ‘holes’ in terms of connections within or between levels in the hierarchy of organization.

7.2.1.1 Properties to Consider

Before we expose the inner workings of the knowledgebase we need to establish some general properties of memory systems that are important for such systems to be useful. Recent research into human memory deviations from these properties has shown how unreliable it is, for example, in witness testimony in court. We seek a memory system that is more reliable than human memory. We pay attention to these properties and check our implementations in digital forms to ensure that our knowledge of systems is reliable and useful.

7.2.1.1.1 Completeness

A knowledgebase is only useful when the elements of the system are complete. What exactly does this mean? We have already seen that deep knowledge depends on a process of decomposition that reveals the lowest level of structural and functional details needed to model any system (or subsystem). This property cannot be guaranteed unless the transparent-box analysis of any subsystem, at any level of organization, is, itself, complete. This means discovering of all components in a system and identifying all of the relevant interactions (i.e. flows) that constitute the interactions between those components. The recursive decomposition of a system, as we have seen, does not lead to an infinite recursion since we can always find components that need no further decomposition. However, failure to push the analysis further down the organization tree when decomposition is warranted will lead to incompleteness of analysis and thus errors in a sufficient description of the system of interest.

---

with the RDBMS structure already worked out. The use of a network database system will be the subject of future research into implementing a knowledgebase.

7.2.1.1.2 Correctness

This should be obvious, but unfortunately is sometimes taken for granted in many current forms of “systems” analysis. The claim made here is that following the procedures of system analysis prescribed in Chapter 5 essentially forces the correctness of the data captured and recorded in the knowledgebase. A main tool for checking correctness is very similar to how a compiler checks syntax in a program. There are several mechanisms that can be built into the data capture and knowledgebase mechanisms to assure this.

7.2.1.1.3 Consistency

One such mechanism is to continually check the consistency of data entered. This can be enforced in several ways. For example, flows from sources to sinks can be checked for consistency. The output of a source has to equal the inputs to all receiving subsystems interfaces (or the sum of all receipts has to match the outputs of all sources). This is easily verified as the analysis proceeds (for example using flow graph theory). The basic rules of consistency for real systems is based on the mass and energy balance laws (e.g. Conservation laws) in physics. If a subsystem is asserted to receive more input than the sum of source outputs permits there is clearly something wrong. The data capture system can alert the analyst that a mass or energy balance equation has been violated and force a reevaluation of the flow.

7.2.1.1.4 Non-duplication

A problem in the decomposition of extremely complex systems is that entities identified at one level of organization might play multiple roles. For example, a source entity, with respect to a focus SOI, might also be a sink entity for other flows. The knowledgebase has to be able to handle such dual roles without confusion. A single entity that is, say, the source of a material flow may also be a sink for a message flow. The knowledgebase system has to be able to distinguish these differences while still recognizing the unity of the entity involved. We will provide examples of how this is accomplished.

7.2.1.1.5 Well-formedness

A well-known problem in computer languages (and mathematics) is the form of statements or sentences have to conform to rules of form in order to be computable. Every computer language is based on a strict syntax to assure these rules are followed. Program compilers are designed to enforce said rules by throwing exceptions when the rules are violated.

Our system language is no different in that it is a formal language even though it is meant to address ‘natural’ systems descriptions. We establish a ‘minimal’ set of rules for how various elements of the language are construed so as to be computable (i.e. be supported by algorithmic representations). At the same time, we allow some more “free-form” constructions that provide flexibility and expressibility, making the language relate to natural languages.

Identification tags, names, type codes, and formulas must conform to a set of standards so that cross-checking (e.g. for non-duplication) can be computationally efficient. After all, the knowledgebase is contained within a computational framework. These items are 'keys' that allow us to access additional information, such as descriptions and histories that are not necessarily as constrained. Examples of well-formed expressions will be provided below. And their linkage to less constrained data will also be shown.

7.2.1.2 The Schema

We use the formal definition of a system from Chapter 3 into a database design called a "schema." In relational database terms this constitutes the set of tables and data elements. Figure 7.2 provides an overview of the basic DBMS schema that we use to implement the intent of Equation 3.1 in Chapter 3.

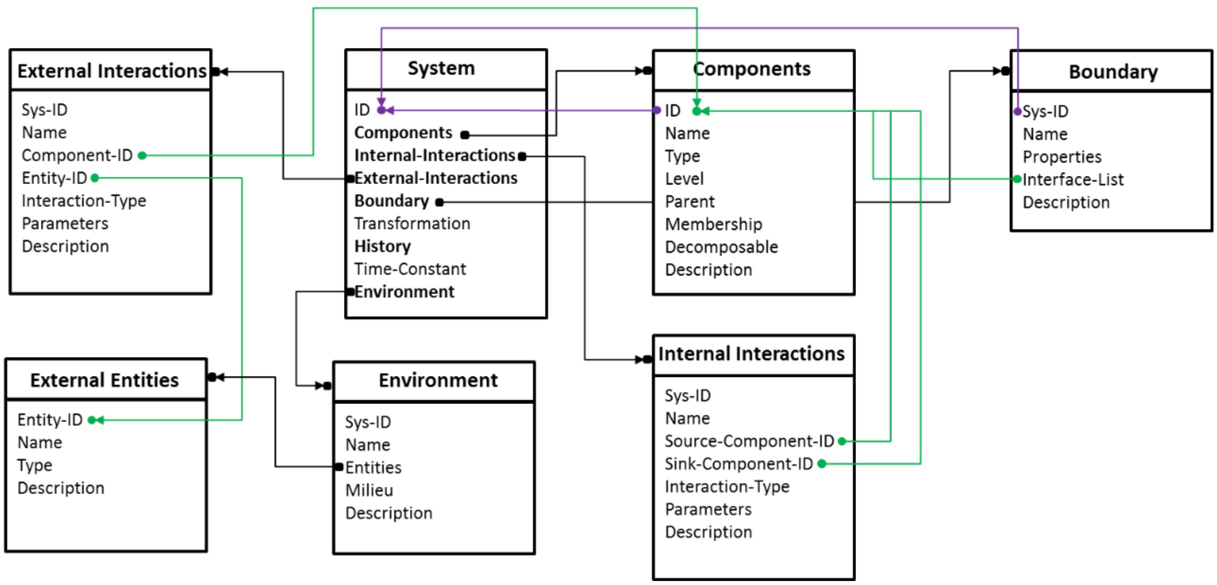

Fig. 7.2. The RDBMS tables associated with storage of system knowledge are shown with some of the relevant links. Black arrows show table links, green arrows show internal data links, and purple arrows show identification links (secondary indexes).

This figure varies somewhat from the way an actual DBMS schema would be implemented but this is in order to show the relations between database tables. Terms shown in bold typeface represent links to other tables. For example, the data element **Components** in the System table represents the logical linkage between the System record and a corresponding record in the Components table following the results of Equation 3.3, components at one level can be subsystems themselves and therefore treated as systems as the analysis proceeds. Various RDBMS products have some variations on their syntax as to how these linkages are made. What we are attempting to show here is a syntax-free, logical diagram of such.

This schema assumes a table for each of the components (sets and graphs) defined in Equation 3.1. What we will now do is describe how the relations are distributed across these tables. Note that in order to make practical implementations of Equation 3.1, we will need to add some additional relations. For example, the inclusion of environmental entities has to be addressed through a set of tables, **Environment** and **External Entities** that will provide the linkages needed to implement Equation 3.5, regarding the graph labeled ***G***.

Additionally note that the database schema is only a “backbone” structure for capturing system knowledge. In the examples below, we will run into examples of other data objects that are better represented in forms other than RDBMS data elements. These will be links to external objects such as pages in a wiki-like form.

7.2.1.2.1 The System-Component-System Relation

Recall from Equation 3.2 that every further decomposable component of a system, $c_{i,j} \in C_i$, is considered a subsystem by Equation 3.3, meaning it is also a system and therefore a member of the System table, as mentioned above. The **System-Component** tables provide this recursive relation. Any component system has an ID that reflects its membership as a subsystem of a parent system. So, for example, a component with an ID of 0.1.2 is also system 0.1.2 and since it is considered decomposable will be found in the **System** table with that ID. The parent systems, 0.1 will, of course, also be present in the **System** table. The main difference between items in the **System** table and the **Components** table is that the former only includes links to items in Equation 3.1, whereas the latter includes specific data items needed to fully specify the component.

In this scheme one can look up an entity as a system or as a component of a higher-level system. This relation establishes the organizational tree described in Chapter 2 (see Figure 2.5).

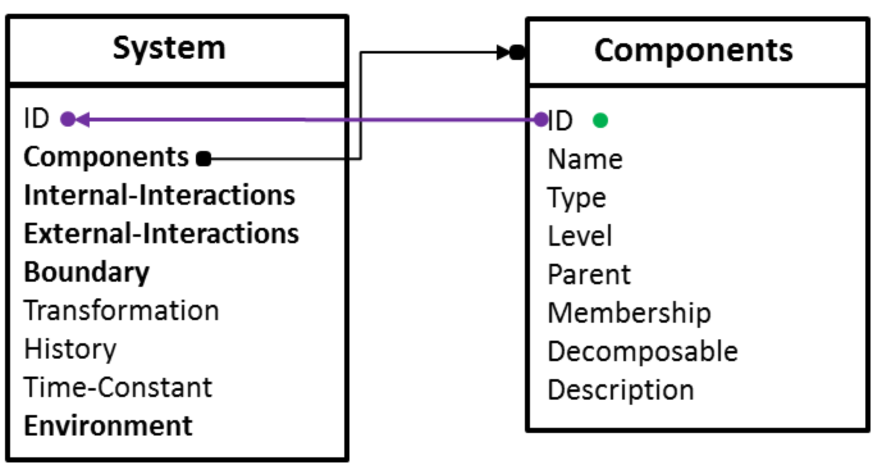

Fig. 7.3. The relation between a system and its component subsystems (Equation 3.2) is captured in these two tables. See the text for details of the relations.

The **System** table contains data elements that directly reflect Equation 3.1. The fields shown in bold (in Figure 7.3) are links to specific tables containing data from each of the elements of

Equation 3.1. The other fields contain either simple data elements, such as ID, or links to non-database documents such as wiki-like web pages. For example, the *Transformation* data element could be a link to a computer program. The *History* data element, while being an element of Equation 3.1, is different. It could be a link to a file of time series data or a set of wiki-like documents containing the historical records of the system. We will have more to say about this element.

The **Components** table contains the set of components, both subsystems and atomic components captured in the system. The two tables are linked by the identification field (ID). An SOI is given the ID integer 0. Its relevant record in the **Components** table is also 0 and provides the top-level information about the whole system. However, in addition, the **Components** table includes the set of subsystems (0.1, 0.2, 0.3,…0.*n*) as well as all of their children. Since every decomposable subsystem is, itself, a system, it will have a corresponding record in the **System** table providing the linkages to the children nodes. Figure 7.4 shows an example of the subsystems of an SOI as they would be entered into the **Components** and **System** tables. The SOI, ID 0, the root of the system tree, appears as a first component in the **Components** table in order to capture its relevant descriptive data. Then, the **Components** table contains all of the subsystems (those indicated as decomposable). However, the same subsystem appears in the System table so that its Equation 3.1 information can be established. Components that are non-decomposable (leaf nodes in the system tree) are not represented in the **System** table since they are not considered as subsystems. They are only represented in records in the **Components** table.

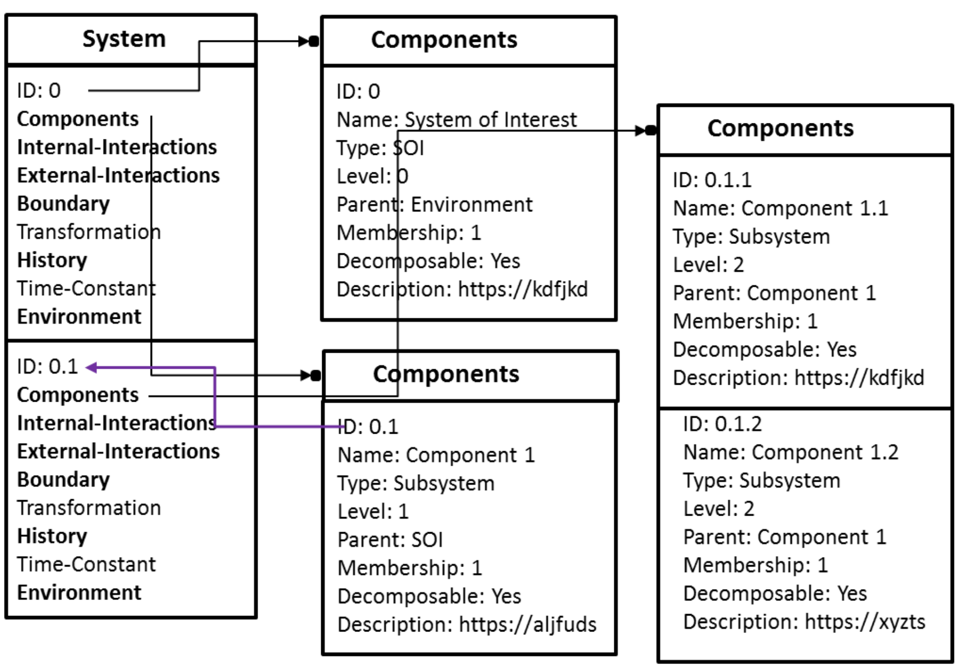

Fig. 7.4. An example of the relation between the System table and Components shows how an SOI is represented along with a subset of its first level of component subsystems.

The implementation of Equation 3.6 in a database schema is shown in Figure 7.5. Every system in the **System** table has a defined boundary object. The **Boundary** table contains the data associated with the boundary from the equation, namely the set of properties, such as fuzziness and permeability, and a list of **Interfaces** that transfer flows into and out of the system.

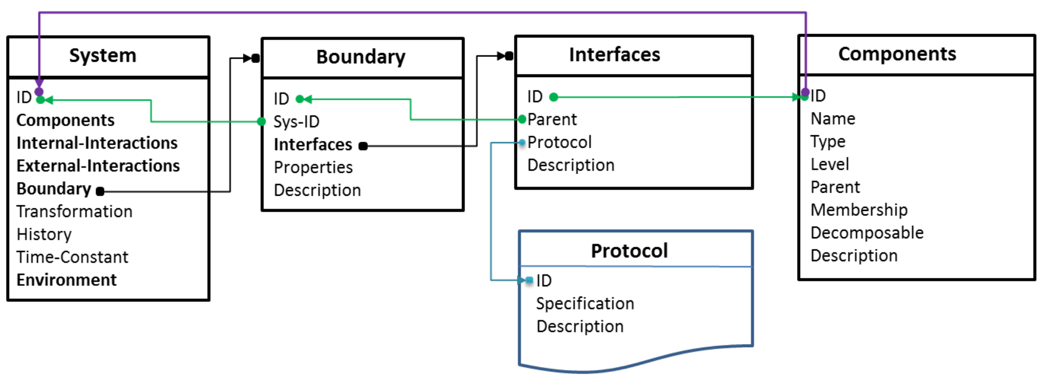

Fig. 7.5. Every system has a boundary object that is comprised of a set of properties and a list of interfaces (Equation 3.6). Every interface is also a component of the system. And since every component subsystem is also a system the linkage returns from the component back to the system.

All interfaces are components (of a special kind) and so the majority of details are found in their component record in the **Components** table. In turn, the component record will be found in the **System** table (purple arrow in the figure) as in Figure 7.5. Note, however, that an interface, as described in Chapter 3, has associated with it a **Protocol** or a special kind of function that is given by a set of operations and timing specifications. These specifications are not necessarily describable within an ordinary database data element and so the **Protocol** data in the **Interface** table will generally point to an external object that is more flexible (e.g. a wiki page or a computer algorithm). Also note that protocols are highly reusable objects. That is, the same protocol may be utilized by many different interfaces in a boundary or even among many different boundaries of other systems. Therefore, these objects may best be kept in the wiki-like part of the system knowledgebase. That is why the shape of the **Protocol** object is a “document” rather than a table. A great deal of research is needed to assess the most appropriate method for representing and storing protocols, but it is clear that this is an achievable goal within the knowledgebase framework.

7.2.1.2.2 History

The methods for capturing and storing the history of a system will vary from system type to type. For example, the history element of an elemental atom (e.g. a hydrogen atom) would most likely be NULL; that is no history is recorded. Histories only start to become important for systems complex enough to have memories of prior states. This is the case for living systems and artifacts designed to record prior states, so must at least be accounted for in our schema. Figure 7.6 shows a generalized approach to how a knowledgebase can include the provision for a history of the system of interest. The recording of the records of history may be contained in another database or may be represented in other kinds of documents (as shown in the figure). In the simplest case, the history is a single file containing state data at discrete time intervals (similar to what has been described for the input-output data records used to analyze the opaque-box scenarios). But the issue of what constitutes the 'history' record of a system is still a very much open question. The schema we have proposed here only provides a placeholder for capturing the history object. We suspect that efficient computer-based approaches will follow that model.

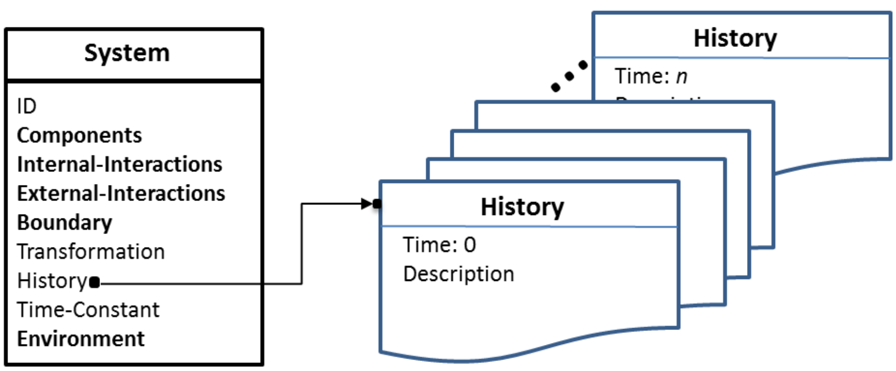

Fig. 7.6. The history element of Equation 3.1 is captured in a series of documents.

7.3 Details of Relations/Tables

In this section we describe some of the tables envisioned to implement the structure of the system definition framework of equations 3.1 and 3.2. Additional tables implement the other equations (structures) defined in Chapter 3, section 3.3.3. Below, an asterisk next to a table field means the contents are actually pointers to other tables or fields implementing the network structure of the system and as shown in Figure 7.2. Other field contents are generally things like the unique IDs of other components or the parent that is the supra-system entity. Here we will only focus on the System Table and the Component Table (Figure 7.3 above in section 7.2.1.2.1) to demonstrate more details of what can be found in the tables. With these examples it should not be hard for a database systems designer to organize the other tables and set up the necessary linkages. The System and Component tables implement Equations 3.1 and 3.2.

7.3.1 System Table

The core table in the knowledgebase schema is the **System** Table. This table contains the embodiment of Equation 3.1 with references to **Components**, **Boundary**, **Internal-Relations**, **External-Relations** (graphs), **History**, and **Environment** tables. These latter tables contain further references and actual data regarding the types of data as specified in Chapter 3.

7.3.1.1 ID

All objects in the knowledgebase have unique identification numbers. This is the relation key, no two systems can have the same **ID**. This applies to all subsystems as well. We use the dotted numeral notation to indicate this key. For example, the SOI is given the singular integer 0 as its **ID**. See below, however, for considerations of how a current SOI may become a subsystem in a larger supra-system structure.

A component of the system ($c_{i,j}$ in Equation 3.2) shows up again in this table with a dotted **ID** such as 0.1 or 0.8.13. That is, this one table will contain the basic information of every subsystem. This is the consequence of Equation 3.2 defining the recursive decomposition structure of a system and the treatment of each subsystem as either an atomic component or a further decomposable system in its own right. See the next section, **Component** table.

7.3.1.2. Name

The name of a system will depend on the disciplinary domain. The name can be in any spoken language, of course, so the range of possibilities is essentially endless. As currently envisioned the name chosen by the lead analyst should be as generic as possible within the domain. Note that if the system is within a type category, e.g. a dog being a type (or kind) of a mammal, then that relation will be handled in the **Component** table.

Naming conventions will likely evolve over time; we see this already in the other sciences and engineering practices. We are not, at this stage, recommending any particular convention.

7.3.1.3 Components*

This field points to a table containing the components of the system. In turn, every decomposable component points back to an element in the **System** table. Again, this is the consequence of Equation 3.2. The **Components** table will contain the data relative to the component while the **System** table treats the components as if they were systems. The fields of the **Components** table are described below.

7.3.1.4 Internal-Interactions*

The **Internal-Interactions** table represents the graph of relations, in Equation 3.4, replicated here: $N_{i,l} = \langle C_{i,l}, L_{i,l} \rangle$. $C_{i,l}$ is really just the set of components identified in equations 3.1 and 3.2. $L_{i,l}$ are the linkages between internal components with other internal components. The schema for this table (Figure 7.2) contains the two components that are interacting along with type of interaction data. For example, if the interaction type is a flow of material one component will be the source and the other the sink. And the **Parameters** element will point to another table (not shown) containing the attributes of the substance that flows. This latter table contains the 'augmentation' data that describes things like flow rates and measurement parameters, such as mass or current, as appropriate to the type of flow (or force).

7.3.1.5 External-Interactions*

As shown in Figure 7.2 external interactions involve one component in the system with an entity in the environment. As with internal interactions we need to identify either the entity or the component as source or sink, when the interaction is a flow or force. The entities are stored in the **External-Entities** table, which serves the same function for keeping track of the entities in the environment as the **Components** table serves for the internals of the SOI. The external interactions implement the tri-partite graph in Equation 3.5, replicated here:

$$G_{i,l} = \langle (C'_{i,l}, Src_{i,l}), (C''_{i,l}, Snk_{i,l}), F_{i,l} \rangle.$$

7.3.1.6 Boundary*

Equation 3.6 provided for the description of a boundary in terms of a set of properties and a list of interfaces embedded in the boundary: $\boldsymbol{B}_{i,l} = \langle P_{i,l}, I_{i,l} \rangle$. The properties of a boundary, in terms of its permeability, fuzziness, etc. can be contained in an augmented wiki page. Its exact format is still the subject of research. The list of interfaces, however, is just a list of pointers back to the **Components** table elements since every interface object is a component of the system. See Figure 7.5.

7.3.1.7 Transformation†

The transformation object is one of several types such as a set of differential equations or a computer program that embodies the way in which a set of inputs to the system (or component subsystem) is transformed into the outputs. A **Transformations** table provides an entry for each component within the system (from subsystems down to atomic components). A pointer in the table entry then points to an appropriate form for computing the transformation. Shown in Figure 7.7 is a typical method for storing a computer program (probably the most common form today) that accomplishes the transformation. This program is written in a language that is suitable for incorporation into an overall simulation, say as a function call. The figure depicts the program for an SOI (e.g. a main() function in C). The structure makes no necessary commitment to which programming language or computational environment (e.g. Mathematica or similar problem solver environment). That will be determined by the simulation environment chosen for running models (see Chapter 9).

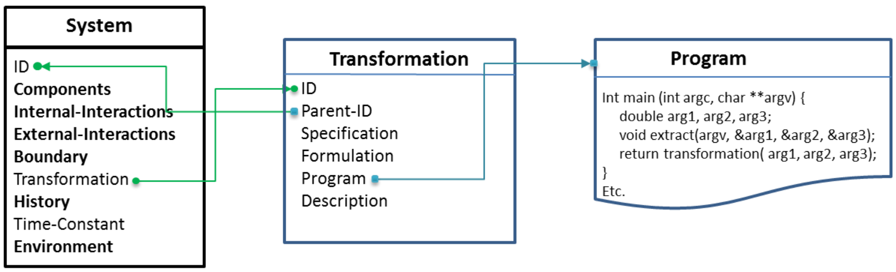

Fig. 7.7. Transformation may be captured in various forms depending on the kind of model/simulation to be derived. Here we show the relation for a computer program that produces the output values based on current inputs.

7.3.1.8 History*

As indicated previously, the history 'object' is the most complicated aspect of the knowledgebase. This is partly because different kinds of systems employ different kinds of history recordings but also because this is a very open research area. Figure 7.6 above suggests a structure in which we consider a time-series recording of states of the system. This can take many forms and so there should not be a rush to standardize what that should be. For relatively simple systems we can specify a time-series recording of states, e.g. a living cell's input/output recordings in an opaque-box analysis. What we should record for a social system of humans is much more open to consideration. In the next chapter we will examine the human social system subsystem – the economy – and provide some examples of what should be considered as relevant to historical documents that can be analyzed for trends and making predictions.

In general, however, it is envisioned that the history objects will be recorded in one or more of the various auxiliary data structures (e.g. Wiki, or time series files) as appropriate to the system.

7.3.1.9 Time-Constant‡

Time is perhaps the most contentious of all characteristics to be applied to system dynamics. What is the best time constant to be used in describing the behavior of a system? There are many factors that need to be taken into consideration to set an appropriate Δt value. From the standpoint of the knowledgebase, the value selected will apply to all components identified in the current level of organization. The selected value has to be small enough in order to capture the minimal change that can be measured in the relevant parameter.

7.3.1.10 Environment*

The environment consists of all of the entities with which the system interacts, i.e. sources and sinks, but also may include unknowns that create a 'milieu'. The pointer contained in this field points to the **Environment Table**.

7.3.2 Components Table

The Component Table contains the detailed data pertaining to a component of a system, including the system itself which has **ID** C0. That is, this **ID** points back to the top-level system in the **System** table. The latter contains a list of pointers to components in this table and the ID points back to the same entity treated as a system. This combination of forward and backward references is what implements the relation between Equations 3.1 and 3.2. These references allow for queries to start in either table and quickly resolve to the proper relation.

7.3.2.1 ID

This is basically the same as in the System table in terms of format and content.

7.3.2.2 Name

This field is redundant for all elements where the component is actually a subsystem of the parent system. If the element is an atomic component, however, the entry will not be found in the **System** table so will have a unique name not found there.

7.3.2.3 Type

The typing of components as subsystems is still an area of much debate. A typology for systems in general was presented in Chapter 2. As envisioned in the current context type categories would be standardized but kept generic. For example, using the Simple, Complex, Complex Adaptive, and Complex Adaptive & Evolvable as the overarching category would be useful in guiding further analysis because those categories tell the analyst what sorts of subsystems should be found at the next level of decomposition. However it might then be necessary to introduce something like sub-types to differentiate more specific KIND-OF relations. Again, we expect this standardization to evolve with practice.

7.3.2.4 Level

This is, of course, the level in the organization hierarchy. The SOI being level 0 and subsequent subsystems being at level 1, and so on.

7.3.2.5 Parent

This is a pointer to the supra-system in which this component participates. For example, if the ID of this component were C0.1.3, then the level would be 2 and the parent pointer would be to C0.1.

7.3.2.6 Membership

This entry contains the specific membership function of the component vis-à-vis the participation degree in the parent system. Recall that membership functions relate to fuzzy systems in which the component may be a member only part of the time and each component may have its own such function.

In the case of a component that has multiple memberships in different parent systems, e.g. a human being can be a member of their workplace, their home, their church, or any number of other systems (admittedly quite complex, but specifiable for averages) but when resident in one system are not resident in the others simultaneously, then the same component may have as many copies as there are memberships, but only one function per copy.

7.3.2.7 Decomposable

This is a simple binary switch indicating if the component is a subsystem (YES) or an atomic component (NO).

7.3.2.8 Description

This field could contain a hyperlink to a Wiki page containing as extensive a text description as is thought necessary to provide a useful description of the entity. What is useful will depend on the context and the considerations of the analysts. The advantage of using a Wiki here is not unlike the Wikipedia usefulness.

7.4 Auxiliary Data Storage and Access

In Chapter 3, section 3.2.2 we made the argument that communication involves a variety of modalities, auditory, visual, etc., shown in Figure 3.1 that are integrated. The knowledgebase concept promoted here supports the idea that knowledge itself needs to be multi-modal.

7.5 Generating Dynamic Models

As stated at the beginning of the chapter, knowledge is useful only when it can be used to consider the future of the world and the system. The static structure of the knowledgebase captures only the elements of the system definition (as applied to a specific physical system). It contains knowledge of the **Transformation** object applicable to each element, but these are just 'equations' or computer code. The real value of the knowledgebase is the ability to generate a dynamic model of the system for simulation. Knowledge is used to anticipate future states of the world (environment + system).

In this section we consider how dynamic models (for computer simulation) may be generated directly out of the knowledgebase and then used to test alternative future behaviors of the system under varying future scenarios of the environment.

References

Alkon, DL (1987) *Memory Traces in the Brain*, Cambridge University Press, Cambridge UK

Boulding, KJ (1956) "General Systems Theory – The Skeleton of Science", *Management Science*, 2, 197, Reprinted in Klir, 2001

Hawkins, J, with Blakeslee, S (2004) *On Intelligence*, Times Books/Henry Holt and Company, New York

Klir, GJ (2001) *Facets of Systems Science*, Second Edition, IFSR International Series on Systems Science and Engineering, Vol. 15, Kluwer Academic Press/Plenum Publishers, New York

Losos, JB (2017) Improbable Destinies: Fate, Chance, and the Future of Evolution, Riverhead Books, New York

Squire, LR & Kandel, ER (2009) *MEMORY: From Mind to Molecules*, Roberts and Company Publishers, Greenwood Village, CO